Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Unidade Curricular de Bases de Dados

Ano Lectivo de 2018/2019

Pizzaria Zé do Cartaxo

Bruno Veloso (78532), Jaime Leite (80757), João Marques (81826) e Nuno Rei (81918)

Janeiro, 2019

BD

Data de Recepção	
Responsável	
Avaliação	
Observações	

Pizzaria Zé do Cartaxo

Bruno Veloso (78532), Jaime Leite (80757),

João Marques (81826) e Nuno Rei (81918)

Janeiro, 2019

Resumo

O relatório apresentado é referente à criação e desenvolvimento de uma base de dados relacional que organiza a informação relativa a uma pizzaria (Pizzaria Zé do Cartaxo), permitindo aos utilizadores autorizados (empregado e dono) consultar esta informação sempre que necessário.

Na fase inicial deste desenvolvimento, foi feito o levantamento e análise de requisitos, com o objetivo de descrever da melhor maneira possível o funcionamento de uma pizzaria. A partir disto, foi mais prático para o grupo criar o MC.

Numa fase consecutiva, e depois de desenvolvido o modelo anteriormente referido, foi feita a passagem para o ML. Para verificar se este era ou não válido, analisaram-se as três formas normais e procedeu-se às alterações necessárias de modo a que estas formas fossem respeitadas na totalidade.

Seguidamente a isto e já numa fase final, através do MySql, foi desenvolvido o Modelo Físico e respetivo povoamento da base de dados. Criaram-se também os utilizadores que podem ter acesso à BD, bem como se especificaram as estruturas a que cada um destes está autorizado a consultar. Codificaram-se “views”, “procedures”, “functions”, “triggers”, “transactions” …de forma a responder ao que era referenciado nos requisitos e fazendo com que a base de dados conseguisse responder de uma maneira acertada às exceções que poderiam surgir.

Em todas as partes referidas anteriormente, o grupo pretendeu implementar da forma mais correta possível os conhecimentos aprendidos nas aulas teóricas e práticas da unidade curricular de Bases de Dados.

Área de Aplicação: Criação e desenvolvimento de uma base de dados

Palavras-Chave: Bases de Dados Relacionais, MySQL, SQL, NoSQL, Neo4J, Grafo, Entidades, Atributos, Relacionamentos, Modelo Físico, Modelo Conceptual, Modelo Lógico.

Índice

Anexos

Índice de Figuras

Índice de Tabelas

1. Introdução

Neste capítulo, será realizada a introdução do projeto a desenvolver, bem como a sua contextualização, caso de estudo, motivação, objetivos e no final é feita a estrutura do relatório.

1.1 Contextualização

Este projeto será realizado no âmbito da cadeira de Bases de Dados.

Por todo o país, existem várias pizzarias. O sucesso de cada uma não depende unicamente de um dado pizzeiro ou do simples empregado de balcão, mas também do SBD que permita o fácil e cómodo acesso a tudo aquilo que seja relevante para o funcionamento de uma pizzaria.

Assim sendo, o grupo pretende neste projeto criar e desenvolver um SBD que arquive informação relativa ao sistema pizzaria.

1.2 Apresentação do Caso de Estudo

Nos dias de hoje, as pizzarias são muito procuradas. Dito isto, as pizzarias precisam de expandir, não só uma expansão das suas infraestruturas, mas também com uma evolução dos seus softwares, neste caso a base de dados por eles usada.

A PZdC é uma das que procura esta expansão a nível do seu software. Para tal acontecer é preciso um SBD que organize informações relativamente à sua pizzaria, de forma a se poder aceder a informação lá guardada de forma organizada e rápida.

Neste projeto pretendemos então criar esta base de dados, que permita à pizzaria do Zé do Cartaxo uma melhor organização.

1.3 Motivação e Objetivos

Dada a quantidade de dados relativa a uma pizzaria que é necessário guardar, torna-se imprescindível conseguir definir uma forma eficiente para manusear e guardar estes dados. Por conseguinte, o grupo pretende criar uma base de dados que consiga armazenar tal conjunto de informação, não só a referente aos materiais e empregados da pizzaria, mas também aos clientes.

Definiremos os requisitos necessários, satisfazendo os que são exigidos pela unidade curricular como também os apresentados neste relatório.

1.4 Estrutura do Relatório

O relatório apresentado tem a seguinte estrutura:

Dedicatórias, seguido de uma apresentação do tema a desenvolver, bem como a apresentação do caso de estudo e o que levou o grupo a trabalhar no tema referido.

Depois, levantamento e análise de requisitos, e também a sua subdivisão por três camadas diferentes: requisitos de descrição, requisitos de exploração e requisitos de controlo.

De seguida, análise ao MC desenvolvido, identificação e caracterização das entidades, dos relacionamentos, das associações entre os atributos com as entidades e os relacionamentos e também a apresentação e consequente explicação do diagrama ER.

Desenvolvimento do ML e sua validação através da normalização, com as interrogações do utilizador e também com as transações estabelecidas.

Passagem do ML para o modelo físico, com a ajuda do MySQL. Apresentação das de queries, transações e de vistas desenvolvidas em SQL.

No fim uma conclusão, análise crítica ao trabalho, anexos e bibliografia.

2. Levantamento e Análise de Requisitos

Nesta secção é apresentada a análise dos requisitos para a elaboração deste trabalho, bem como o método adoptado para o levantamento dos mesmos.

2.1 Método de levantamento e de análise de requisitos adotado

De forma a obter uma melhor percepção do problema em mãos, foi observado o normal funcionamento da pizzaria e feitas entrevistas com os empregados da mesma. Foi então obtido um melhor entendimento quanto à necessidade desta base de dados e foi feita uma listagem de requisitos para a realização da mesma.

2.2 Requisitos levantados

Apresentam-se de seguida os requisitos referentes à PZdC.

Numa fase anterior à definição do MC da base de dados apresentada neste relatório, o grupo pretendeu descrever da forma mais sucinta e sólida possível as funcionalidades da base de dados da Pizzaria, desde a introdução de um simples produto na base de dados até ao pagamento de um pedido por parte de um dado cliente.

2.2.1 Requisitos de descrição

Deve ser guardada informação quanto a empregados, pedidos, clientes, fornecedores, ingredientes e pedidos de ingredientes.

Para cada empregado deve ser guardado o seu número de identificação, o seu nome, email, telefone, NIF e local onde mora.

Para cada cliente deve ser guardado o seu número de contribuinte, nome, data de nascimento, local onde mora, email caso tenha e pretenda fornecer, e os seus números de telefone/telemóvel.

Para cada fornecedor de ser guardado o seu nome (nome da empresa) e seu número de identificação.

Para cada pedido deve ser guardado o seu número de identificação, o número do cliente a que corresponde, o empregado que fez o pedido, o preço do pedido e a data em que foi feito.

Para cada ingrediente deve ser guardado o seu número de identificação, nome do ingrediente, o número de ingrediente em stock, o seu preço de venda e compra e o fornecedor que nos fornece esse ingrediente.

Para cada pedido de ingrediente deve ser guardado o número de identificação do pedido a que corresponde, o ingrediente que foi pedido, o preço de venda deste ingrediente e a quantidade pedida.

2.2.2 Requisitos de exploração

Deverá ser possível:

1. Ver uma lista de pedidos ordenada por preço e depois por data que foram efetuados;
2. Ver um catálogo com os preços dos ingredientes ou produtos da pizzaria, ordenando por preço;
3. Ver qual o fornecedor que fornece mais a pizzaria;
4. Ver o TOP 5 ingredientes mais pedidos;
5. Ver o número total de pedidos de um cliente num dado intervalo de tempo;
6. Atualizar o stock de um ingrediente;
7. Ver os ingredientes contidos num determinado pedido;
8. Ver os contactos de um cliente;
9. Ver o empregado que serviu mais vezes um determinado cliente;
10. Ver os ingredientes que um dado fornecedor fornece;
11. Verificar o stock de um ingrediente;
12. Ver o total gasto por um cliente na pizzaria.

2.2.3 Requisitos de controlo

Para os requisitos de controlo foram estabelecidos dois perfis.

O perfil *admin* para o dono da Pizzaria (Zé do Cartaxo) que terá acesso a todas as operações sobre a base de dados.

O outro é o perfil *empregado* para ser utilizado pelos empregados da Pizzaria, estes podem aceder, alterar e inserir informações relacionadas com clientes e pedidos, podem aceder e alterar informações relativas aos ingredientes e apenas consultar as informações relacionadas com os fornecedores e empregados.

2.3 Análise geral dos requisitos

O cliente e empregado são as principais entidades deste sistema. O cliente é identificado pelo seu número de contribuinte e o empregado é identificado por um número que é dado pelo admin.

Aos pedidos estão associados o cliente tal como o empregado.

O cliente pede ingredientes, o empregado anota no sistema o pedido total e vê se consegue efetuar o pedido.

A um pedido só existe associado um empregado e um cliente.

Quando precisamos de ingredientes, um fornecedor fornece-nos o que precisamos. O admin ou empregado atualiza o stock.

3. Modelação Conceptual

A seguir apresenta-se o MC desenvolvido pelo grupo, bem como a análise aos seus diversos componentes, como sendo as entidades e respetivos atributos, bem como os relacionamentos presentes.

3.1 Apresentação da abordagem de modelação realizada

Para o desenvolvimento do MC, foi importante para o grupo ter os requisitos já definidos. A partir daí, foram identificadas as entidades, os atributos e seus tipos, e as relações entre atributos e entidades. Para fazer a ligação entre as várias entidades, escolheram-se as chaves candidatas e primárias.

Como especificado no tópico 3.7 deste relatório, foi necessário verificar que o MC correspondia de forma ordeira ao que era proposto para a base de dados.

3.2 Identificação e caracterização das entidades

Apresenta-se de seguida uma tabela desenvolvida com o intuito de identificar as entidades que compõem o MC. Nesta tabela, apresenta-se a descrição das entidades, bem como respectivos sinónimos e ocorrências no sistema.

Nome da entidade	Descrição	Sinónimos	Ocorrências
Pedido	Elemento principal da pizzaria, constituído por vários ingredientes	Encomenda	Um pedido é feito por um cliente e introduzido no sistema por um empregado
Ingrediente	Elemento que chega à pizzaria através de um fornecedor e que é escolhido pelos clientes para os seus pedidos	Produto	Um ingrediente chega à pizzaria, numa certa quantidade, através de um fornecedor e fica em armazém até ser utilizado num pedido
Fornecedor	Quem distribui os vários ingredientes à pizzaria	Provedor	Um fornecedor distribui uma dada quantidade de vários ingredientes para posteriormente serem introduzidos num pedido
Cliente	Pessoa que escolhe os ingredientes que fazem parte do seu pedido	Freguês	Um cliente faz um pedido que fica registado no sistema
Empregado	Quem regista os pedidos	Funcionário	Um empregado insere os pedidos no sistema

Tabela 1: Entidades do modelo conceptual.

3.3 Identificação e caracterização dos relacionamentos

Seguindo a análise de requisitos, bem como as entidades envolventes, definimos da seguinte forma o relacionamento e cardinalidades entre entidades:

Nome da entidade	Multiplicidade	Relacionamento	Multiplicidade	Nome da entidade
Empregado	1	Realiza	N	Pedido
Pedido	1	É feito por	1	Empregado
Cliente	1	Faz	N	Pedido
Pedido	1	É feito por	1	Cliente
Pedido	1	Contém	N	Ingrediente
Ingrediente	1	Está contido	N	Pedido
Ingrediente	1	Fornecido por	1	Fornecedor
Fornecedor	1	Fornece	N	Ingrediente

Tabela 2: Relacionamentos entre as Entidades.

3.4 Identificação e caracterização das associações dos atributos com as entidades e relacionamentos

Depois de identificados e caracterizados os relacionamentos e entidades, tendo sempre em conta os nossos requisitos, passamos à atribuição de atributos às entidades e relacionamentos.

3.4.1 Atributos simples/compostos

Depois de analisar os requisitos, verificamos que eram necessários alguns atributos simples nas entidades, este serão elaborados mais abaixo. Não foram usados quaisquer atributos compostos.

3.4.2 Atributos derivados

Mais uma vez, depois de analisar os requisitos reparamos que necessitamos de atributos derivados, no caso da entidade “Pedido”, este atributo é o “Preço” e é derivado porque depende do que o “Cliente” peça como ingredientes para completar o seu “Pedido”. Também temos outro atributo derivado que pertence à entidade “PedidoIngrediente” e o atributo é “Preço” uma vez que depende do ingrediente e da quantidade do ingrediente que o cliente escolha.

3.4.3 Atributos multivalor

Depois de analisar os atuais requisitos, também verificamos que precisávamos de um atributo multivalorado, que no nosso caso era o “Contacto”, que pertence à entidade “Cliente” e remete para os contactos de um cliente, no qual um cliente pode ter um ou mais números de telefone/telemóvel.

3.4.4 Atributos de relacionamento

Após a observação do levantamento de requisitos, reparamos que para certos requisitos seriam necessários atributos de relacionamento que são “Quantidade” e o “Preço”, em que o atributo “Quantidade” representa a quantidade que foi pedida de certo ingrediente e o atributo “Preço” representa o preço deste ingrediente pedido vezes a quantidade pedida.

3.4.5 Associação entre atributos e entidades

Uma vez encontrados os atributos necessários, vamos passar aos seus detalhes. Apresentamos uma tabela para cada entidade com os detalhes necessários de cada atributo:

Nome da Entidade	Atributo	Descrição	Tipo	Tamanho (Bytes)
Empregado	Id	Número de identificação do empregado	INT	4
	Nome	Nome do empregado	VARCHAR(45)	46
	Email	Email do empregado	VARCHAR(45)	46
	Telefone	Telefone do empregado	VARCHAR(9)	10
	NIF	NIF do empregado	INT	4
	Localidade	Local onde mora do empregado	VARCHAR(20)	21

Tabela 3: Tabela da Entidade Empregado.

Nome da Entidade	Atributo	Descrição	Tipo	Tamanho (Bytes)
Cliente	Nr_contribuinte	Número de contribuinte do cliente	INT	4
	Nome	Nome do cliente	VARCHAR(45)	46
	DataNascimento	Data de nascimento do cliente	DATE	3
	Localidade	Local onde mora o cliente	VARCHAR(20)	21
	Email	Email do cliente	VARCHAR(45)	46

Tabela 4: Tabela da Entidade Cliente.

Nome da Entidade	Atributo	Descrição	Tipo	Tamanho (Bytes)
Fornecedor	Id	Número de identificação do fornecedor	INT	4
	Nome	Nome da empresa que fornece	VARCHAR(45)	46

Tabela 5: Tabela da Entidade Fornecedor.

Nome da Entidade	Atributo	Descrição	Tipo	Tamanho (Bytes)
Ingrediente	Id	Número de contribuinte do cliente	INT	4
	Nome	Nome do ingrediente	VARCHAR(45)	46
	Stock	Stock disponível do ingrediente	INT	4
	PrecoCompra	Preço que custa comprar ao fornecedor	DECIMAL(8,2)	5
	PrecoVenda	Preço a que vendemos o ingrediente	DECIMAL(8,2)	5
	Id_fornecedor	Número de identificação do fornecedor	INT	4

Tabela 6: Tabela da Entidade Ingrediente.

Nome da Entidade	Atributo	Descrição	Tipo	Tamanho (Bytes)
Pedido	Id	Número de identificação do pedido	INT	4
	Id_cliente	Número de identificação do cliente que pede	INT	4
	Id_empregado	Número de identificação do empregado que regista o pedido	INT	4
	Preco	Preço total do pedido	DECIMAL(8,2)	5
	Data	Data do pedido	DATE	3

Tabela 7: Tabela da Entidade Pedido.

Nome da Entidade	Atributo	Descrição	Tipo	Tamanho (Bytes)
Relacionamento: Pedido-Ingrediente	Id_pedido	Número de identificação do pedido	INT	4
	Id_ingrediente	Número de identificação do ingrediente	INT	4
	Preco	Preco total da quantidade do ingrediente	DECIMAL(8,2)	5
	Quantidade	Quantidade do ingrediente	INT	4

Tabela 8: Tabela do Relacionamento entre as Entidades Pedido e Ingrediente.

Nome da Entidade	Atributo	Descrição	Tipo	Tamanho (Bytes)
ClienteContacto	Id_cliente	Número de identificação do cliente	INT	4
	Telefone	Número de telefone do cliente	VARCHAR(9)	10

Tabela 9: Tabela do atributo multivalorado ClienteContacto.

3.5 Chaves primárias, candidatas e alternativas

A informação presente no SBC assenta sobre cinco entidades: “Pedido”, “Ingrediente”, “Fornecedor”, “Cliente” e “Empregado”.

A entidade “Pedido” tem como chaves candidatas “Id”, “Id_Cliente”, “Data”, “Preço” e “Id_Empregado”. Como “Data”, “Preço”, “Id_Cliente” e “Id_Empregado” são más opções, uma vez que pode haver chaves repetidas, resta-nos para chave primária o “Id”, não havendo assim chaves alternativas para esta entidade.

O “Ingrediente” tem como chaves candidatas “Id”, “Nome”, “Stock”, “Id_fornecedor”, “PreçoCompra” e “PreçoVenda”. Uma vez que o “Stock”, “Id_fornecedor”, “PreçoCompra” e “PreçoVenda” podem ter valores iguais, faz com que estas chaves sejam más opções. Resta assim duas chaves candidatas que é “Id” e “Nome”, o “Nome” podia ser chave principal porque não há nomes iguais de ingredientes, mas o grupo decidiu escolher “Id” como chave principal porque é mais fácil para comparar chaves em int do que em varchar, ficando assim “Nome” como chave alternativa.

O “Fornecedor” tem duas chaves candidatas que são o “Id” e “Nome”, ambas podiam ser chaves principais porque o “Id” não se repete e também não há nomes de empresas iguais, mas o grupo decidiu que o “Id” era uma melhor decisão porque é mais fácil de comparar int com int, ficando assim “Nome” como chave alternativa.

O “Cliente” tem como chaves candidatas “Nr_contribuinte”, “Nome”, “Email”, “Localidade”, “Contacto” e “DataNascimento”. Como “DataNascimento”, “Nome” e “Localidade” podem ter valores iguais, estes são descartados de chaves candidatas. Entre “Nr_contribuinte”, “Email” e “Contacto” temos que mais uma vez a decisão fica entre o que é mais fácil comparar, neste caso é o “Nr_contribuinte” que fica como chave primária, pelo que as chaves alternativas são “Contacto” e “Email”.

A entidade “Empregado” tem como chaves candidatas “Nome”, “Id”, “Email”, “Telefone”, “NIF” e “Localidade”. Como “Nome” e “Localidade” podem ter valores iguais, estes são discartados de chaves candidatas. Entre “Email”, “Telefone”, “NIF” e “Id”, o grupo escolheu “Id” porque, entre atributos que sejam int que são neste caso o “Telefone”, “NIF” e “Id”, o “Id” é quem vai ter o menor valor de todos para comparação, tornando mais fácil comparar, logo “Id” é a nossa chave primária e chaves alternativas são o “Email”, “Telefone” e “NIF”.

3.6 Apresentação e explicação do diagrama ER

Uma vez que já foram apresentadas muitas explicações das nossas escolhas sobre o MC, relativamente a entidades e relacionamentos, faremos um breve resumo de todo o diagrama ER, assim como a sua explicação superficial, pois já se abordaram certos tópicos anteriormente.

Na imagem abaixo, podemos ver que o relacionamento de “Empregado” e “Pedido” é de 1:N, uma vez que um empregado pode efetuar vários pedidos. Por sua vez, “Pedido” e “Empregado” têm relação de 1:1, visto que um pedido pode ser efetuado por um empregado.

O relacionamento de "Ingrediente" e "Fornecedor" é de 1:1, uma vez que um ingrediente só corresponde a um determinado fornecedor. No entanto, a relação “Fornecedor” e “Ingrediente” já é de 1:N uma vez que um fornecedor tem mais do que um tipo de ingrediente.

Seguidamente analisa-se os relacionamentos entre as entidades “Cliente” e “Pedido” e “Pedido” com “Cliente”. O primeiro destes é de 1:N, uma vez que um cliente faz mais que um pedido. Por sua vez, o segundo relacionamento já é de 1:1 uma vez que um determinado pedido só é feito por um cliente.

Por fim, a relação de “Pedido” e “Ingrediente” é de 1:N nos dois sentidos uma vez que um pedido por conter vários ingredientes e um ingrediente pode estar em num ou mais pedidos.

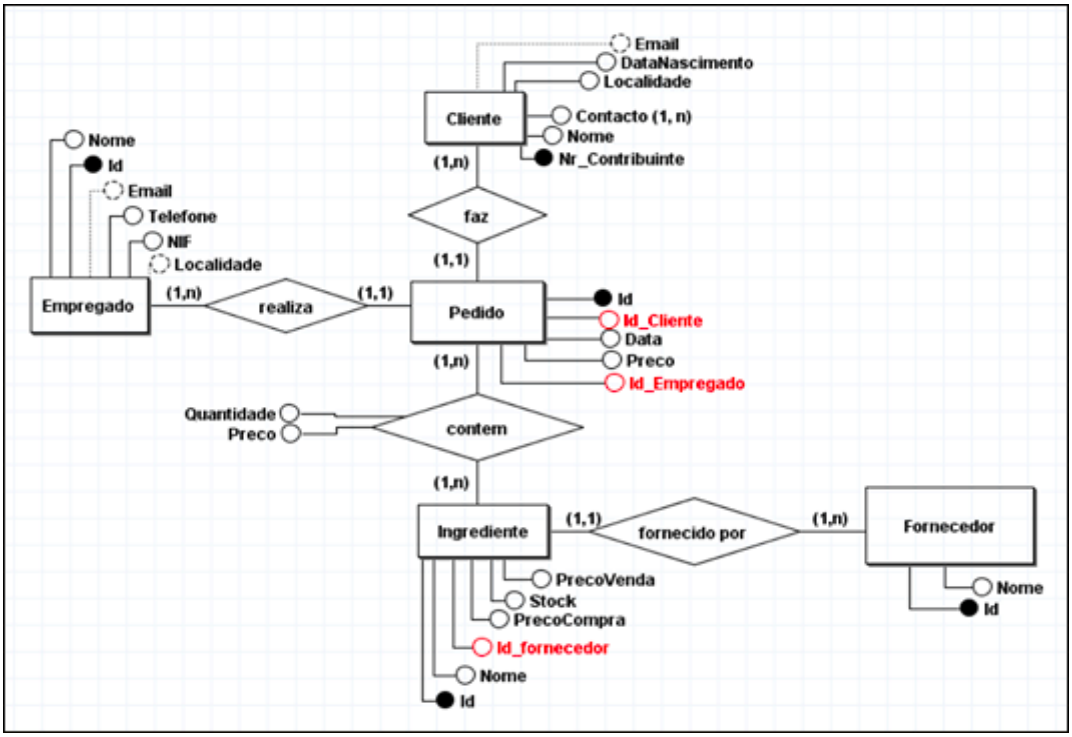


Fig 1: Modelo Conceptual.

3.7 Validação do modelo de dados com o utilizador

Depois de tudo decidido relativamente ao MC, vamos ver se realmente com as nossas escolhas conseguimos responder a todos os pedidos que o Zé do Cartaxo nos propôs.

Passaremos à análise de *query* em *query*:

- ★ **Ver uma lista de pedidos ordenada por preço e depois por data que foram efetuados.**

  Para responder a esta *query,* é necessário a entidade "Pedido" e seus atributos "Preço" e "Data", uma vez que ficou ao nosso critério a implementação então podemos incluir também a entidade "Cliente" e "Empregado" ligando assim o atributo "Id_cliente" de "Pedido" ao atributo "Nr_contribuinte" de cliente e sabemos o nome do cliente, e também podemos ligar "Id_empregado" de "Pedido" ao "Id" de "Empregado" e sabemos o nome do empregado, logo consegue-se responder com sucesso.

- ★ **Ver um catálogo com os preços dos ingredientes ou produtos da pizzaria, ordenando por preço.**

  Para esta *query,* é necessário a entidade "Ingrediente" e seus atributos "Preço" e "Nome", também conseguimos ligar o fornecedor através do atributo "Id_fornecedor" de "Ingrediente" ligando com o atributo "Id" da entidade "Fornecedor", logo consegue-se responder com sucesso.

- ★ **Qual o fornecedor que fornece mais a pizzaria?**

  Conseguimos responder usando as entidades "Ingrediente" e "Fornecedor", uma vez que a entidade "Ingrediente" tem um atributo "Id_fornecedor" conseguimos juntar com a entidade "Fornecedor" através do atributo "Id" e assim saber qual fornecedor tem mais ingredientes seus na pizzaria.

- ★ **Quais os 5 ingredientes mais pedidos?**

  Para esta *query,* juntamos as entidades "Ingrediente" e "PedidoIngrediente" através do atributo "Id_ingrediente" de "PedidoIngrediente" e "Id" de "Ingrediente" e através do atributo "Quantidade" de "PedidoIngrediente" conseguimos saber o TOP 5 ingredientes mais pedidos.

- ★ **Qual o número de total de pedidos de um cliente num determinado intervalo de tempo?**

  Para responder a esta *query,* usamos as entidades "Pedido" e "Cliente", comparando o atributo "Nr_contribuinte" de "Cliente" e "Id_Cliente" de "Pedido", e comparando a "Data" de "Pedido" conseguimos calcular o número de pedidos de um cliente num intervalo de tempo específico.

★ **Atualizar o stock de um ingrediente.**

Para atualizar o stock de um ingrediente basta ter o “Id” de “Ingrediente” e a “Quantidade” a introduzir ou a tirar, atualizando assim o stock.

★ **Quais os ingredientes contidos no pedido X?**

Para responder a esta *query,* precisamos das entidades “Ingrediente” e “PedidoIngrediente” e como “PedidoIngrediente” tem um atributo “Id_pedido” conseguimos procurar no pedido certo e depois como temos também em “PedidoIngrediente” um atributo “Id_ingrediente” comparamos com “Id” de “Ingrediente” e saber o nome do ingrediente através do atributo “Nome” em “Ingrediente” e assim responder com sucesso a esta *query* retornando todos os ingredientes de X pedido.

★ **Quais os contactos de um cliente?**

Como temos os clientes identificados por “Nr_contribuinte” é só procurar o valor que desejamos da pessoa e retornar o atributo “Contacto” para responder com sucesso esta *query*.

★ **Qual o empregado que serviu mais vezes X cliente?**

Precisamos das entidades “Empregado” e “Pedido”, juntando a informação do atributo “Id_empregado” de “Pedido” e “Id” de “Empregado” conseguimos saber nome deste, e juntando também “Id_Cliente” de “Pedido” com o valor que quisermos procurar e comparando quem atende mais vezes este cliente conseguimos saber responder a esta *query* com sucesso.

★ **Quais os ingredientes que X fornecedor fornece?**

Precisamos da entidade “Fornecedor” para saber o nome deste através do atributo “Nome”, e tendo a entidade “Ingrediente” que tem um atributo “Id_fornecedor” que se compara com o fornecedor que queremos procurar, depois é só juntar com o “Id” de “Fornecedor” para saber o nome e conseguimos responder a esta *query*.

★ **Qual o stock de X ingrediente?**

Basta a entidade “Ingrediente” que tem como atributo “Id” para comparar com o ingrediente que queremos procurar e uma vez que “Ingrediente” tem o atributo “Stock” conseguimos saber o stock de um determinado ingrediente com sucesso.

★ **Quanto foi que X cliente gastou na pizzaria, durante a vida toda?**

Para responder a esta *query*, precisamos da entidade “Pedido” e comparar o atributo “Id_Cliente” com o número que queiramos procurar e somamos os preços dos pedidos deste.

Concluímos assim, depois de uma análise profunda e da validação com o utilizador, que o nosso modelo responde a todas as *queries* e que é então aceite.

4. Modelação Lógica

4.1 Construção e validação do modelo de dados lógico

Depois de decidido qual o MC a ser implementado, passamos à elaboração do nosso ML. Neste subcapítulo apresentaremos como foi construído e validado o nosso ML.

4.1.1 Entidades

Uma vez que o atributo “Contacto” do nosso MC era multivalorado, quando implementado no ML, surgirá numa tabela separada da entidade “Cliente”. Gera-se a tabela “ClienteContacto”, que tem uma chave primária composta que é o “Id_cliente” e “Telefone”, e tem uma chave estrangeira “Id_cliente” para conseguir ligar à tabela “Cliente”. As chaves primárias e estrangeiras serão explicadas mais abaixo, quando se falar no relacionamento entre entidades.

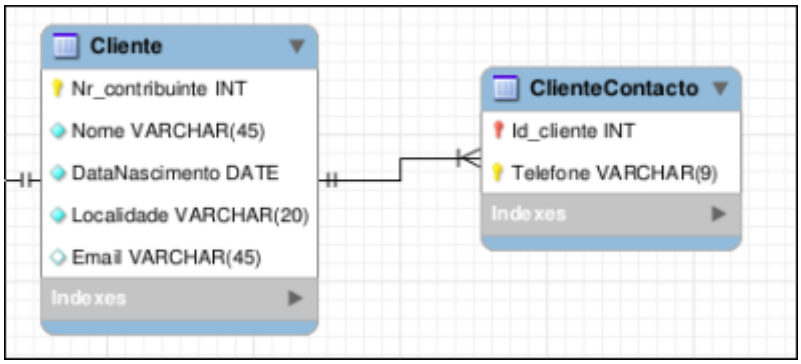


Fig 2: Entidade Cliente e ClienteContacto.

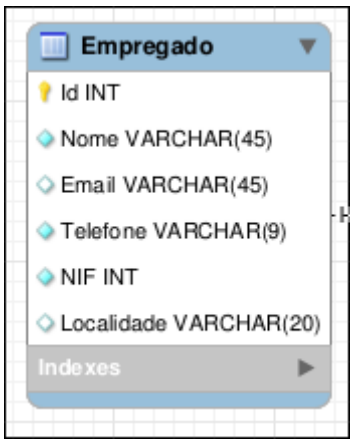


Fig 3: Entidade Empregado.

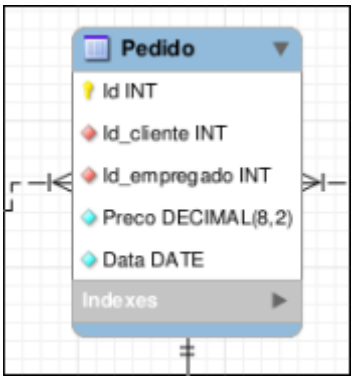


Fig 4: Entidade Pedido.

Fig 5: Entidade Ingrediente.

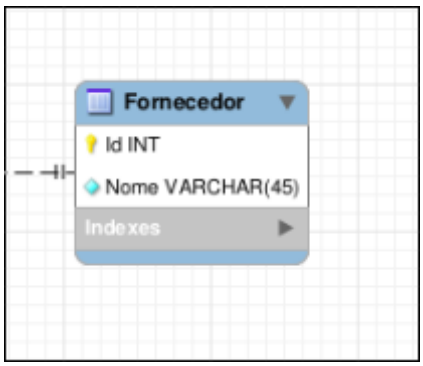


Fig 6: Entidade Fornecedor.

4.1.2 Relacionamentos entre entidades

A relação entre “Empregado” e “Pedido é de 1:N. A chave principal de “Empregado” é o “Id” que identifica o empregado, e a chave principal de “Pedido” é o “Id” que remete para o número de pedido. Uma vez que existe um relacionamento entre “Empregado” e “Pedido” em “Pedido” vai haver uma chave estrangeira de forma a remeter para “Empregado” e esta chave estrangeira é o “Id_empregado”. Também existe uma relação de 1:N entre “Cliente” e “Pedido”, logo também existe uma chave estrangeira em “Pedido” chamada “Id_cliente” que vai remeter para o cliente que faz o pedido.

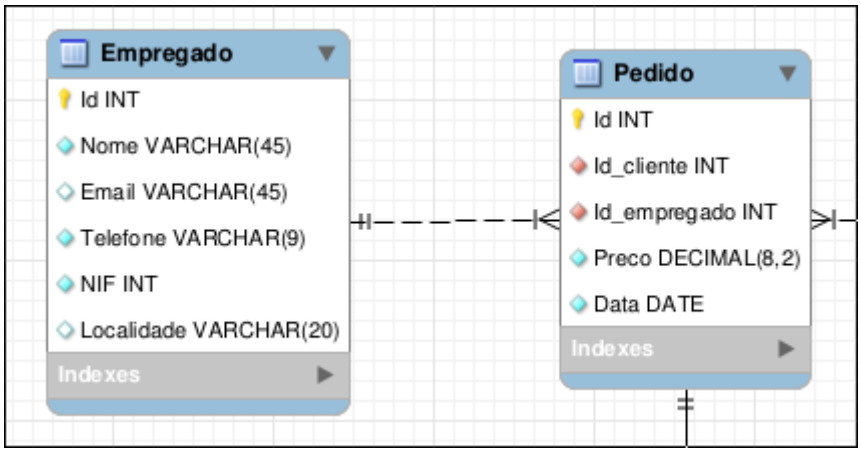


Fig 7: Relacionamento entre Empregado e Pedido.

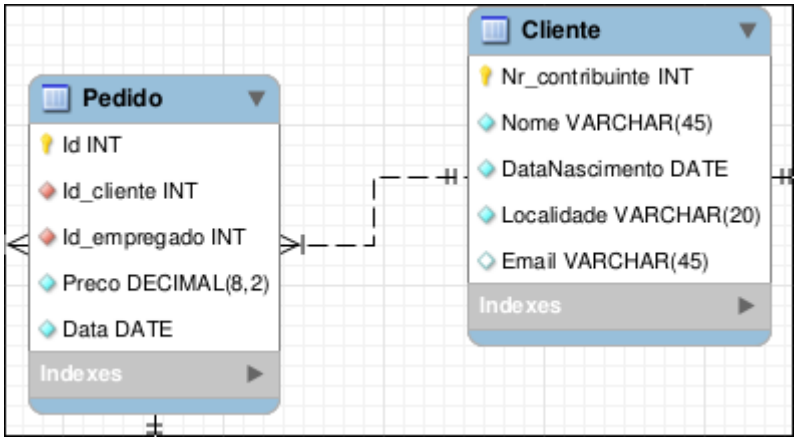


Fig 8: Relacionamento entre Pedido e Cliente.

Um ingrediente chega à pizzaria através de um fornecedor. Por tal razão, faz sentido ligar estas tabelas, em que é a tabela “Ingrediente” que tem uma chave estrangeira que é “Id_fornecedor” pois, caso contrário, ocorreriam repetições nas tabelas, o que fazia com que, numa subsecção posterior, se verificasse que não cumpria primeira forma normal. Já a tabela “Fornecedor” só tem uma chave principal “Id” para saber qual o número de identificação de um fornecedor.

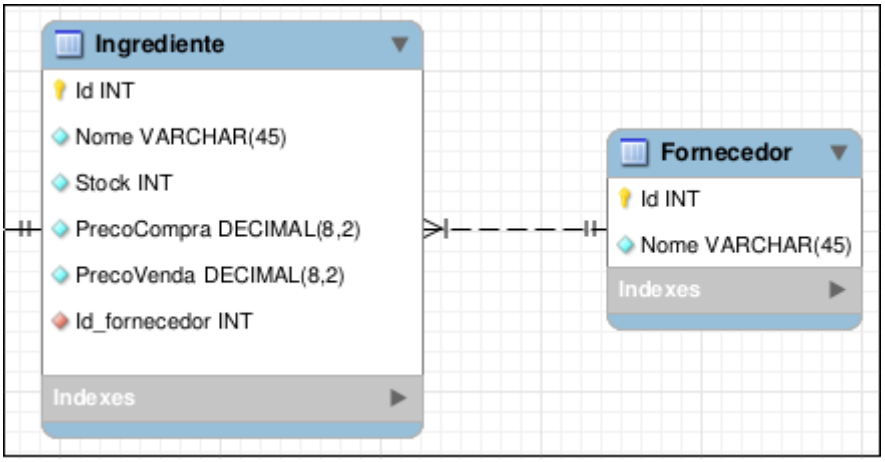


Fig 9: Relacionamento entre Ingrediente e Fornecedor.

Como a relação entre “Pedido” e “Ingrediente” é de N:M, criou-se uma nova tabela com a designação de “PedidoIngrediente”. Nesta estão presentes os identificadores da tabela “Pedido” e da tabela “Ingrediente”, que são o “Id_pedido” e “Id_ingrediente” e cada um deles é uma chave estrangeira, no entanto os dois juntos foram uma chave primária composta. Para além disso, esta nova tabela contém também mais dois atributos: “Preço” e “Quantidade”. O primeiro deles é calculado através da multiplicação da “Quantidade” pelo “PreçoVenda” do ingrediente em questão.

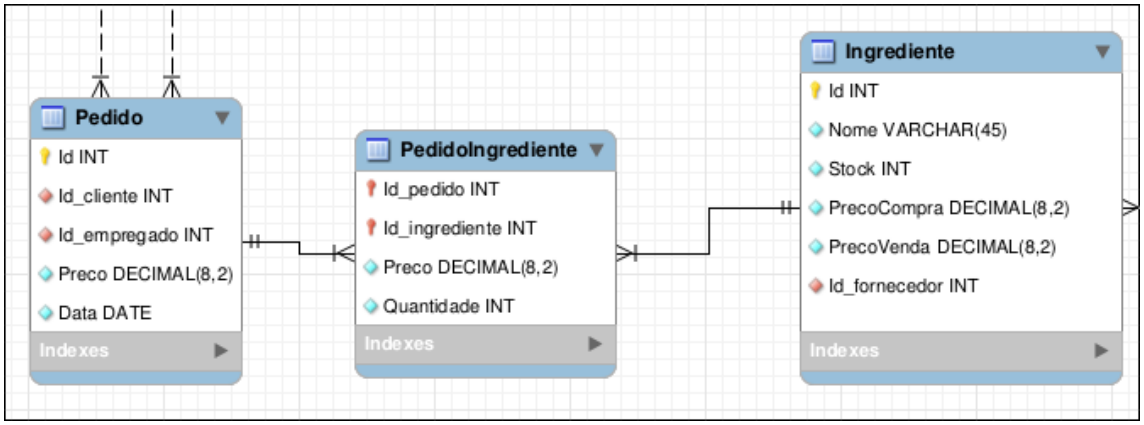


Fig 10: Relacionamento entre Pedido e Ingrediente.

4.2 Desenho do modelo lógico

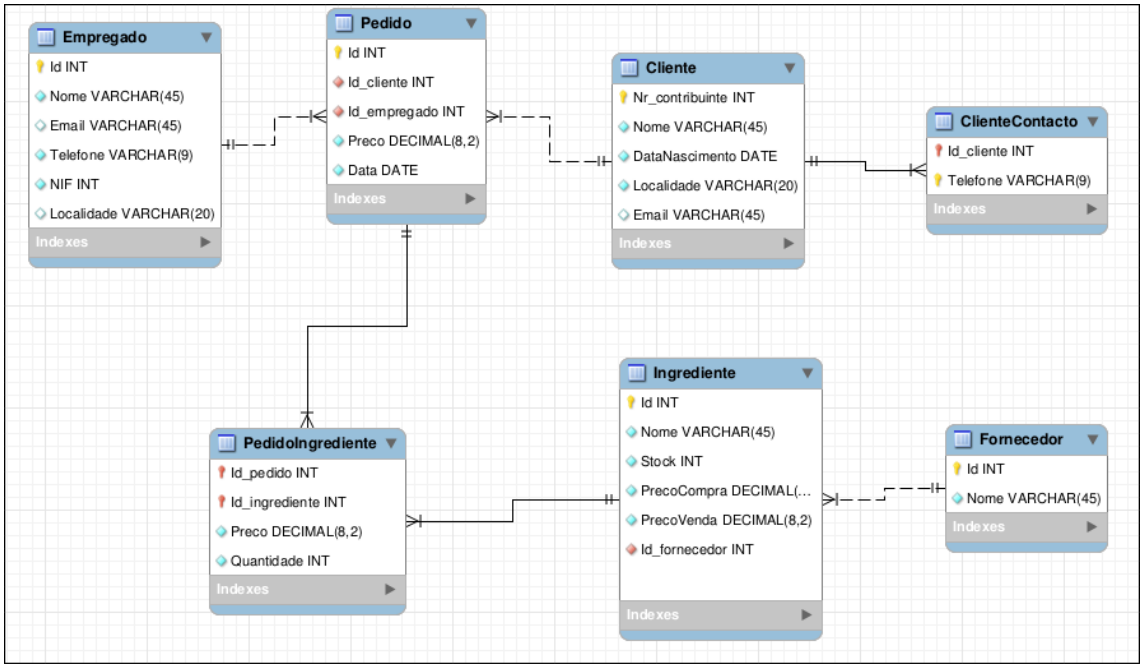


Fig 11: Modelo Lógico.

4.3 Validação do modelo através da normalização

Com o objetivo de aumentar a coesão, coerência e o desempenho da base de dados desenvolvida pelo grupo, bem como a integridade dos dados, é feito de seguida a normalização dos dados, desde a primeira até à terceira forma normal (inclusive).

Primeira forma normal - as tabelas presentes no ML não têm atributos repetidos. No entanto existe um atributo multivalorado, relativo à tabela “Cliente”, os telefones de um cliente. Para não existirem elementos repetidos na tabela referida, criou-se uma tabela designada por “ClienteContacto”, que tem como atributos o id do cliente (chave estrangeira) e um dos telefones desse mesmo cliente. Assim, todas as tabelas cumprem a primeira forma formal.

Segunda forma normal - em todas as tabelas presentes no ML, todos os atributos que não são chave primária apenas se encontram numa única tabela, quando querem ser referenciados noutra tabela usamos a chave primária da mesma. Sendo assim, não tivemos de fazer alteração nas tabelas ou mesmo criar outras para respeitar a segunda forma normal. Consequentemente, ao ser validada a primeira forma normal e verificada também a segunda, as tabelas cumprem a segunda forma normal.

Terceira forma normal - No ML desenvolvido, nenhum dos atributos não chave depende de outros igualmente classificáveis, para uma dada tabela. Não havendo dependências funcionais e sendo cumprida a segunda forma normal,, as relações encontram-se na terceira forma normal.

4.4 Validação do modelo com interrogações do utilizador (alguns exemplos)

Vamos mostrar apenas três interrogações que no sub-capítulo 5.3 serão traduzidas para linguagem *MySQL*.

- Pedidos de um Cliente num dado intervalo de tempo
  Π C.Nome, P.data, P.Preco (
  □(C.nome=nomeCliente □ P.data>=dataInicio □ P.data<=dataFim)
  ((P) ▷◁ P.Id_Cliente=C.Nr_contribuinte (C)))
  P-> Pedido e C->Cliente
- Fornecedor com mais variedade de produtos
  Π F.Nome, COUNT(F.Id) (limit 1(□ COUNT(I.Id_fornecedor)DESC(□ I.Id_fornecedor
  ((I) ▷◁ I.Id_fornecedor=F.Id (F)))))
  F->Fornecedor e I->Ingrediente
- Os 5 ingredientes mais utilizados em pedidos
  Π I.Nome, SUM(PI.Quantidade (limit 5(□ SUM(PI.Quantidade) DESC(□ I.Nome

((I) ▷◁ I.Id=PI.Id_ingrediente (PI))))

I->Ingrediente e PI->PedidoIngrediente

4.5 Validação do modelo com as transações estabelecidas

Através do nosso Esquema Lógico conseguimos perceber que quase todas as transações serão obtidas por uma única operação numa só tabela. Por exemplo a atualização do stock de um ingrediente (Fig 12) é apenas necessário efetuar alterações na tabela Ingrediente, ou criar um novo pedido em que só será criada uma nova tabela Pedido (Fig 13).

A exceção ao mencionado anteriormente é a transação para inserir um ingrediente num pedido. Para que a consistência dos dados na base de dados seja mantida há várias operações que terão de executar como um todo, e de ter o seu sucesso garantido. Quando um ingrediente vai ser inserido num pedido (Fig 14) é necessário efetuar o "lock" do ingrediente a ser inserido, para reduzir o seu stock e obter o preço de venda, criar uma nova tabela PedidoIngrediente, e ainda atualizar a tabela Pedido, atualizar o atributo Preco somando o Preco da PedidoIngrediente.

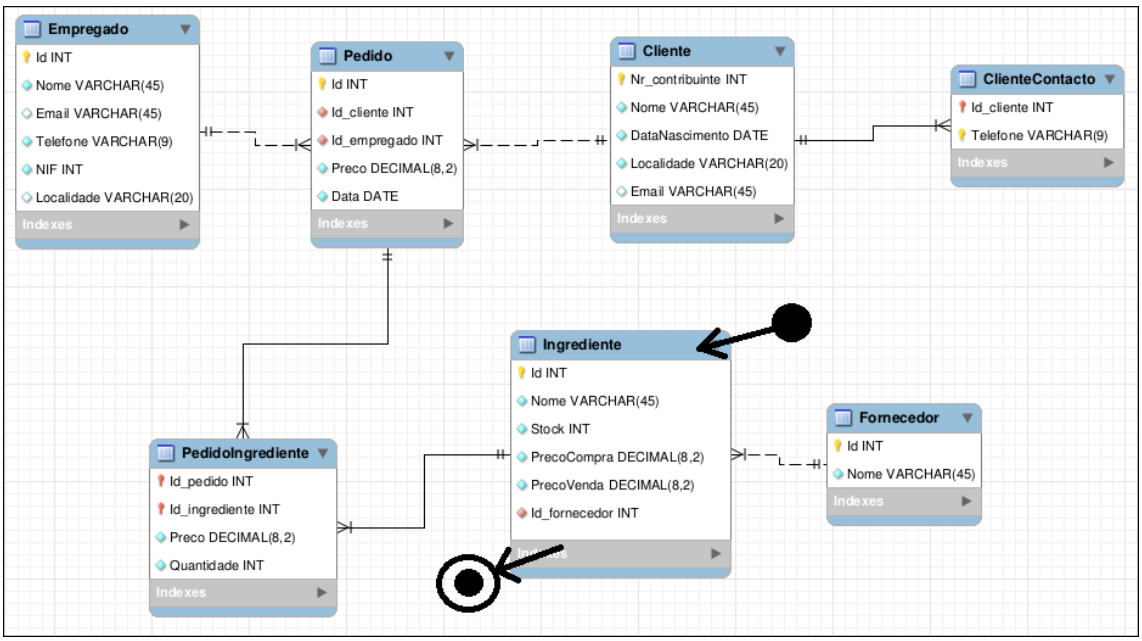


Fig 12: Mapa da transação da atualização de stock.

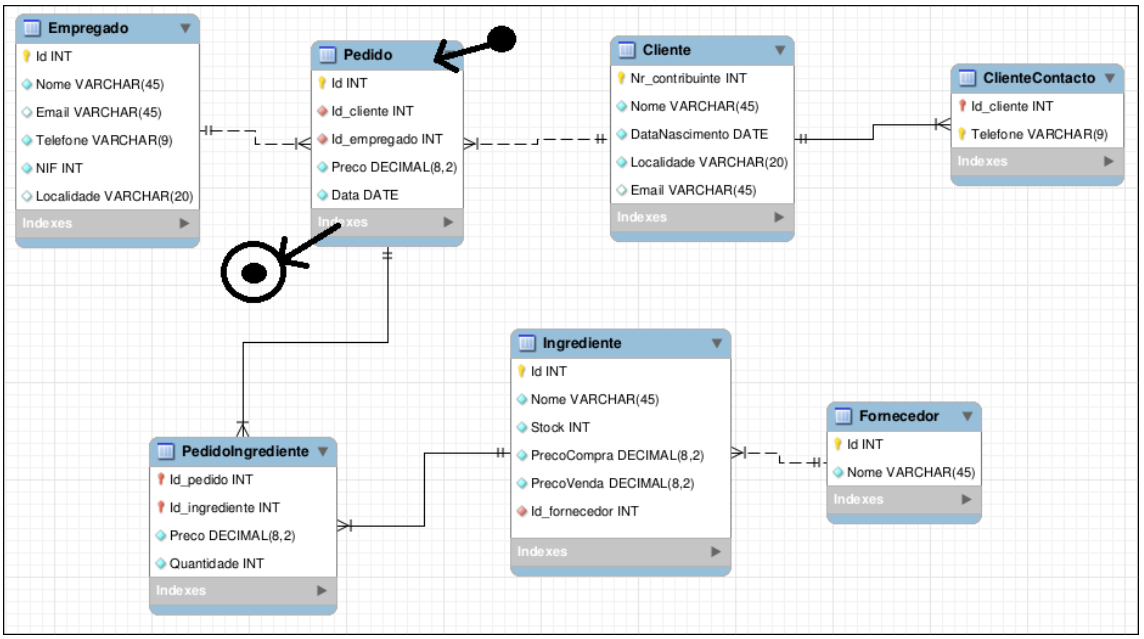


Fig 13: Mapa da transação criar novo pedido.

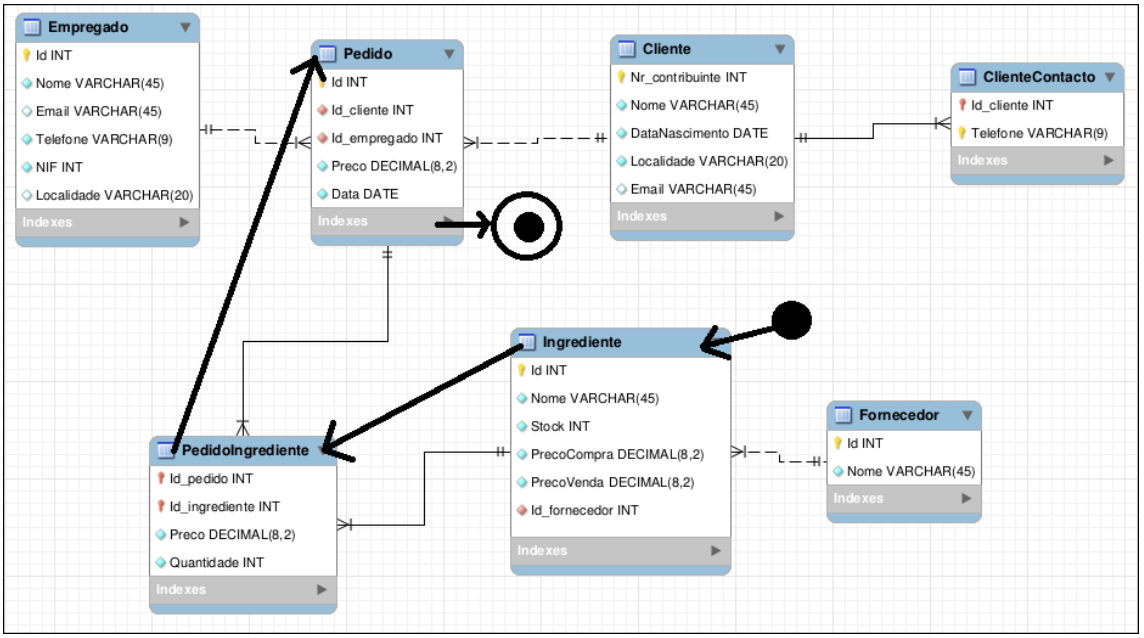


Fig 14: Mapa da transação inserir ingrediente no pedido.

4.6 Revisão do modelo lógico com o utilizador

Após uma análise junto do utilizador conclui-se que o ML desenvolvido pelo grupo aborda de forma correta os conceitos requeridos para o mesmo.

5. Implementação Física

5.1 Seleção do sistema de gestão de bases de dados

No projeto desenvolvido pelo grupo, foi usado o MySQL como sistema de gestão de bases de dados, pois foi o fornecido na unidade curricular de BD para implementação desta base de dados.

5.2 Tradução do esquema lógico para o sistema de gestão de bases de dados escolhidos em SQL

Através do *MySQL Workbench*, nomeadamente da opção *Forward Engineer* que é disponibilizada pelo mesmo, o grupo conseguiu fazer a conversão do ML para o sistema de gestão de bases de dados (Anexo 2).

5.3 Tradução das interrogações do utilizador para SQL (alguns exemplos)

```
DELIMITER $$
CREATE PROCEDURE pedidosCliente(IN nomeCliente VARCHAR(45), IN dataInicio DATE ,IN dataFim DATE)
BEGIN
        SELECT C.Nome, P.data, P.Preco
            FROM Pedido AS P
            INNER JOIN Cliente AS C
                ON P.Id_Cliente = C.Nr_Contribuinte
            WHERE (C.nome = nomeCliente && P.data >= dataInicio && P.data <= dataFim);
END; $$
DELIMITER ;
```

Fig 15: Procedure para obter os pedidos de um cliente num intervalo de tempo.

```
SELECT F.Nome, COUNT(F.Id) AS NrFornecimentos
    FROM Ingrediente AS I
    INNER JOIN Fornecedor AS F
        ON I.Id_fornecedor = F.Id
    GROUP BY I.Id_fornecedor
    ORDER BY COUNT(I.Id_fornecedor) DESC
    LIMIT 1;
```

Fig 16: Query para obter o fornecedor que mais variedade de ingredientes fornece.

```
SELECT Ingrediente.Nome, SUM(PedidoIngrediente.Quantidade)
    FROM Ingrediente
    INNER JOIN PedidoIngrediente On Ingrediente.Id=PedidoIngrediente.Id_ingrediente
    GROUP BY Ingrediente.Nome
    ORDER BY SUM(PedidoIngrediente.Quantidade) DESC
    LIMIT 5;
```

Fig 17: Query para obter o top 5 dos ingredientes mais utilizados.

5.4 Tradução das transações estabelecidas para SQL

Após a análise detalhada do modelo de base de dados percebemos que as transações mais importantes e que com mais frequência irão ocorrer são:

- Criar um novo Pedido;
- Atualizar o Stock de um Ingrediente;
- Inserir um Ingrediente num pedido;

Para a criação de um novo pedido e atualização do stock de um ingrediente, é apenas necessária a realização de uma única operação, dando origem às transações da Fig 17 e Fig 18.

```
DROP PROCEDURE criaNovoPedido;

DELIMITER $$
CREATE PROCEDURE criaNovoPedido (IN clienteID INT, IN empregadoID INT)
BEGIN
    DECLARE novoID INT;
    SET novoID = (SELECT (MAX(Id)+1) FROM Pedido);
    INSERT INTO Pedido (Id, Id_cliente, Id_empregado, Preco, Data)
        VALUES(novoID, clienteID, empregadoID, 0.00, DATE(NOW()));
END;
$$
DELIMITER ;
```

Fig 18: Procedure para criar um novo pedido.

```
DELIMITER $$
CREATE PROCEDURE atualizaStock(IN id INT, IN qtaIntroduzir INT)
BEGIN
    UPDATE Ingrediente
        SET Ingrediente.stock =  Ingrediente.stock + qtaIntroduzir
        WHERE Ingrediente.id = id;
END; $$
DELIMITER ;
```

Fig 19: Procedure para atualizar o stock de um ingrediente.

Para o caso particular da inserção de um ingrediente num pedido, existe a possibilidade de o mesmo ingrediente estar a ser inserido em pedidos diferentes, ou seja um problema de concorrência, havendo então a necessidade de garantir exclusão mútua no processo. Quando é inserido um ingrediente num pedido há também um número de “passos” que devem ser garantidos:

- Criação de uma nova tabela PedidoIngrediente com o Id do pedido, o Id do ingrediente e a quantidade do mesmo;
- Atualizar o atributo Stock na tabela Ingrediente;

- Calcular o preço multiplicando a quantidade do ingrediente pelo precoVenda da tabela Ingrediente e atualizar o atributo Preco da tabela recém criada PedidoIngrediente;
- Atualizar o atributo Preco na tabela Pedido somando o preço calculado no ponto anterior.

Basta que um dos pontos anteriores “falhe” para que a base de dados passe a um estado inconsistente e portanto foi decidido que seria necessário recorrer à transação (Fig 19) para garantir que só serão guardadas as mudanças na base de dados caso todas as operações sejam executadas com sucesso.

```
DELIMITER $$
CREATE PROCEDURE insereIngredienteNoPedido (IN pedidoID INT, IN ingredienteID INT, IN Qt INT)
BEGIN
    DECLARE  Erro BOOLEAN DEFAULT 0;
    DECLARE precoV DECIMAL(8,2);
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET Erro = 1;

    START TRANSACTION;

    INSERT INTO PedidoIngrediente (Id_pedido,Id_ingrediente,Preco,Quantidade)
        VALUES (pedidoID, ingredienteID, 0.00, Qt);
    UPDATE Ingrediente SET Stock = Stock - Qt WHERE Id = ingredienteID;
    SET precoV = (SELECT PrecoVenda
                  FROM Ingrediente AS I
                  WHERE I.Id = ingredienteID);

    UPDATE PedidoIngrediente SET Preco = (PedidoIngrediente.Quantidade*precoV)
        WHERE Id_pedido = pedidoID AND Id_Ingrediente = ingredienteID;

    UPDATE Pedido
            SET Preco = Preco + (SELECT Preco FROM PedidoIngrediente WHERE Id_pedido = pedidoID AND Id_Ingrediente = ingredienteID)
            WHERE Id = pedidoID;
    IF Erro = 1
    THEN ROLLBACK;
    ELSE COMMIT;
    END IF;
END; $$
DELIMITER ;
```

Fig 20: Transaction para inserir ingrediente num pedido.

5.5 Escolha, definição e caracterização de índices em SQL

Uma vez que o número de clientes pode vir a crescer bastante, decidimos recorrer a um índice que facilitasse a procura de um cliente na base de dados. Usamos para este efeito o atributo “Email” da entidade “Cliente”.

```
ALTER TABLE Cliente
ADD INDEX Email(Email);

DROP INDEX Email ON Cliente;
```

Fig 21: Índice para o atributo Email.

5.6 Estimativa do espaço em disco da base de dados e taxa de crescimento anual

No subcapítulo 3.4.5 relativo à associação entre entidades e atributos já foram especificados todos os atributos e o seu tamanho, sendo que o tamanho ocupado por cada linha de uma dada tabela é o seguinte (os valores representam o valor mínimo e máximo):

- Ingrediente: 23 a 68 Bytes por linha, como existem 17 ingredientes o valor máximo ocupado neste momento é 1156 Bytes.
- Pedido: 20 Bytes por linha, como existem 11 pedidos o valor total ocupado é 220 Bytes.
- Empregado: 12 a 131 Bytes por linha, como existem 2 empregados o valor máximo ocupado neste momento é 262 Bytes.
- Cliente: 10 a 120 Bytes por linha, como existem 7 clientes o valor máximo ocupado neste momento é 840 Bytes.
- ClienteContacto: 5 a 14 Bytes por linha, como existem 4 contactos o valor máximo ocupado neste momento é 56 Bytes.
- Fornecedor: 5 a 50 Bytes por linha, como existem 5 fornecedores o valor máximo ocupado neste momento é 250 Bytes.
- PedidoIngrediente: 17 Bytes por linha, como existem 70 registos o valor total ocupado é 1190 Bytes.

No caso do Pedido e PedidoIngrediente como não existem atributos de tamanho variável as linhas terão sempre o mesmo tamanho.

5.7 Definição e caracterização das vistas de utilização em SQL

Esta *view* mostra uma lista de pedidos, em que cada linha aparece o identificador do pedido, o nome do empregado que fez o pedido, o nome do cliente que pediu, o preço total do pedido e a data em que este foi feito.

```
CREATE VIEW viewListaPedidos AS
    SELECT P.Id AS Identificador, E.Nome AS Empregado, C.Nome AS Cliente, P.Preco AS Preco, P.Data AS Data
        FROM Pedido AS P
        INNER JOIN Empregado AS E
            ON P.Id_empregado = E.Id
        INNER JOIN Cliente AS C
            ON P.Id_cliente = C.Nr_contribuinte
        ORDER BY P.Preco DESC, P.Data DESC;
```

Fig 22: View da lista de pedidos ordenada por preço e data.

Esta *view* mostra um catálogo em que aparece o identificador de um ingrediente, seguindo com o nome do ingrediente, o seu preço de venda, o seu fornecedor e o identificador do fornecedor, catalogando esta informação para todos os ingredientes disponíveis.

```sql
CREATE VIEW catalogo AS
    SELECT I.Id AS ID, I.Nome AS Ingrediente, I.PrecoVenda AS Preco, F.Nome AS Fornecedor, F.Id AS Id_Fornecedor
        FROM Ingrediente AS I
        INNER JOIN Fornecedor AS F
            ON I.Id_fornecedor = F.Id
        ORDER BY I.PrecoVenda DESC;
```

Fig 23: View da lista de ingredientes ordenada por preço de venda.

5.8 Definição e caracterização dos mecanismos de segurança em SQL

```sql
/*
Criação de um administrador (dono)
*/
CREATE USER 'admin'@'localhost';
    SET PASSWORD FOR 'admin'@'localhost' = 'admin';
```

Fig 24: Criação do user Admin.

```sql
/*
Criação funcionário
(Criamos 1 funcionario porque neste momento apenas temos um, se for adicionado um novo são lhe dadas as respetivas permissões)
*/
CREATE USER 'func1'@'localhost';
    SET PASSWORD FOR 'func1'@'localhost' = 'func1';
```

Fig 25: Criação do user Empregado.

```sql
GRANT SELECT, INSERT, UPDATE, DELETE ON pizzaria.* TO 'admin'@'localhost';
```

Fig 26: Permissões para o user Admin.

```sql
-- Tabela Cliente
GRANT SELECT, INSERT, UPDATE ON pizzaria.Cliente TO 'func1'@'localhost';

REVOKE DELETE ON pizzaria.Cliente FROM 'func1'@'localhost';
```

Fig 27: Permissões para o user Empregado na tabela Cliente.

```sql
-- Tabela ClienteContacto
GRANT SELECT, INSERT, UPDATE ON pizzaria.ClienteContacto TO 'func1'@'localhost';

REVOKE DELETE ON pizzaria.ClienteContacto FROM 'func1'@'localhost';
```

Fig 28: Permissões para o user Empregado na tabela ClienteContacto.

```sql
-- Tabela Fornecedor
GRANT SELECT ON pizzaria.Fornecedor TO 'func1'@'localhost';

REVOKE INSERT, UPDATE, DELETE ON pizzaria.Fornecedor FROM 'func1'@'localhost';
```

Fig 29: Permissões para o user Empregado na tabela Fornecedor.

```
-- Tabela Empregado
GRANT SELECT ON pizzaria.Empregado TO 'func1'@'localhost';

REVOKE INSERT, UPDATE, DELETE ON pizzaria.Empregado FROM 'func1'@'localhost';
```

Fig 30: Permissões para o user EMpregado na tabela Empregado.

```
-- Tabela Pedido
GRANT SELECT, INSERT, UPDATE ON pizzaria.Pedido TO 'func1'@'localhost';

REVOKE DELETE ON pizzaria.Pedido FROM 'func1'@'localhost';
```

Fig 31: Permissões para o user Empregado na tabela Pedido.

```
-- Tabela PedidoIngrediente
GRANT SELECT ON pizzaria.PedidoIngrediente TO 'func1'@'localhost';

REVOKE INSERT, UPDATE, DELETE ON pizzaria.PedidoIngrediente FROM 'func1'@'localhost';
```

Fig 32: Permissões para o user Empregado na tabela PedidoIngrediente.

Foram implementados dois triggers (Fig 33 e Fig 34) com o intuito de controlar a quantidade de um ingrediente que é inserida num pedido para que esta não exceda o stock existente desse ingrediente e também para que não seja inserida uma quantidade nula de um ingrediente num pedido. Os triggers são ativados antes de um insert e update. Com o auxílio da função implementada temStock (Anexo 4), que serve para obter o stock de um ingrediente, caso se verifique a situação descrita acima não é permitida a alteração na base de dados.

```
DELIMITER $$
CREATE TRIGGER pedidoIngrediente_before_update
    BEFORE UPDATE ON PedidoIngrediente
    FOR EACH ROW
BEGIN
    IF (((temStock(NEW.Id_ingrediente)<NEW.Quantidade) OR NEW.Quantidade <= 0)=TRUE) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Quantidade de ingrediente maior que o stock existente';
    END IF;
END; $$
DELIMITER ;
```

Fig 33: Trigger que aciona antes de fazer update no PedidoIngrediente.

```
DROP TRIGGER pedidoIngrediente_before_insert;

DELIMITER $$
CREATE TRIGGER pedidoIngrediente_before_insert
    BEFORE INSERT ON PedidoIngrediente
    FOR EACH ROW
BEGIN
    IF (((temStock(NEW.Id_ingrediente)<NEW.Quantidade) OR NEW.Quantidade <= 0)=TRUE) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Quantidade de ingrediente maior que o stock existente';
    END IF;
END; $$
DELIMITER ;
```

Fig 34: Trigger que aciona antes de fazer insert no PedidoIngrediente.

5.9 Revisão do sistema implementado com o utilizador

Depois de uma última revisão dada pelo utilizador da base de dados mostrou-se bastante satisfeito com as propriedades do mesmo. Após algumas semanas de utilização fomos contactados pelo Zé do Cartaxo por email mostrando nos toda a sua satisfação com o resultado obtido para a sua pizzaria através do SBD implementado. Podemos ler o que nos foi enviado pelo Sr. Zé do Cartaxo no Anexo 1.

6 Implementação da Base de Dados NoSQL

6.1 Justificação da utilização de um sistema NoSQL

As razões que nos levaram a passar de um sistema SQL para um sistema NoSQL são: ao longo do tempo o número de informações a guardar na base de dados vai crescer demasiado, pelo que num sistema SQL esta quantidade de informação começa a afetar o desempenho das queries, já em NoSQL(base de dados em grafo, no nosso caso) o desempenho mantém-se constante ao longo do tempo. Outra das razões é: em modelo de dados de grafos há uma maior flexibilidade, por exemplo, se quisermos remover ou modificar a base de dados podemos fazê-lo sem pôr em risco o resto das estruturas e funcionalidades, inclusivé as queries são mais fáceis de implementar.

6.2 Identificação e Descrição dos objetivos da Base de dados

O sistema de base de dados NoSQL que será utilizado para desenvolver o resto do trabalho e continuar a responder às queries requeridas pelo dono da pizzaria é baseado em grafos.

Com o desenvolvimento de uma base de dados em Neo4j aproveitamos os benefícios de uma estrutura orientada por grafos. Com este tipo de estrutura, o percurso entre os vários tipos de nodos é mais eficiente do que entre tabelas e as relações entre os nodos é mais simples de explicitar, tirando proveito de um melhor desempenho relativamente ao uso de uma base de dados SQL.

6.3 Identificação e Explicação das interrogações realizadas a Base de Dados NoSQL

As queries que escolhemos para serem implementadas nesta base de dados NoSQL são:

- ★ **Top 5 ingredientes mais pedidos, com respectivo fornecedor**

  Para responder a esta query, necessitamos dos nodos do tipo “Pedido”, “Ingrediente” e “Fornecedor”. Para cada nodo “Ingrediente”, vemos em quantos pedidos está o mesmo. Ordenam-se os ingredientes por ordem decrescente, pela ocorrência referida anteriormente e usa-se o comando “LIMIT 5” para apresentar os cinco ingredientes mais pedidos. Acrescenta-se também o nome do fornecedor dos ingredientes, usando “f.Nome” (em que f é do tipo “Fornecedor”) no comando “RETURN”.

★ **Qual o cliente que mais gastou**

Nesta query, analisamos os nodos do tipo “Cliente” e “Pedido”. Para cada cliente acumula-se os valores das variáveis “p.Preco”, em que p é um Pedido. Faz-se uma ordenação dos clientes a partir do valor acumulado e no final usa-se o comando “LIMIT 1”, para apresentar o cliente que mais gastou.

★ **Qual o cliente com mais pedidos**

Para cada nodo “Cliente” vê-se quantas ligações existem a “Pedido”. Ordenam-se os valores por ordem decrescente, em relação à quantidade de ligações efetuadas por nodo. No final usa-se o comando “LIMIT 1”, para se apresentar apenas o cliente com maior número de pedidos.

★ **Qual o empregado que efetuou mais pedidos**

Na resposta a esta query, usam-se os nodos “Cliente” e “Pedido”. Para cada cliente, são contadas o número de ligações aos nodos do tipo “Pedido”. Ordenam-se os clientes por ordem decrescente, em relação ao número de ligações referido anteriormente. Faz-se o “LIMIT 1” para obtermos o cliente com mais pedidos efetuados.

★ **Quantas vezes um cliente pediu um dado ingrediente distribuído por um determinado fornecedor**

Para esta query, necessitamos dos tipos de nodos: “Cliente”, “Pedido”, ”Ingrediente” e “Fornecedor”. Analisam-se estes nodos e respetivas relações entre os mesmos, sendo feito um filtro ao nome do cliente, nome do ingrediente e fornecedor. Finalmente, é feito um “Return” onde se apresenta o nome do ingrediente pretendido, bem como o número de pedidos em que o mesmo está incluído e, por fim, o nome do fornecedor referido inicialmente.

★ **Qual o lucro da pizzaria num dado intervalo de tempo**

Necessita-se, para esta query, dos nodos do tipo “Pedido” e “Ingrediente”. Para cada pedido, cuja data está limitada, faz-se uma soma dos seguintes valores: diferença entre o preço de compra e venda dos ingredientes multiplicado pela quantidade dos ingredientes que se encontram no pedido. Soma-se os valores resultantes por pedido e obtém-se o lucro da pizzaria.

6.4 Definição da estrutura base para o sistema NoSQL de acordo com requisitos e interrogações

De forma a cumprir todas as queries foi desenvolvida a seguinte estrutura para o sistema NoSQL:

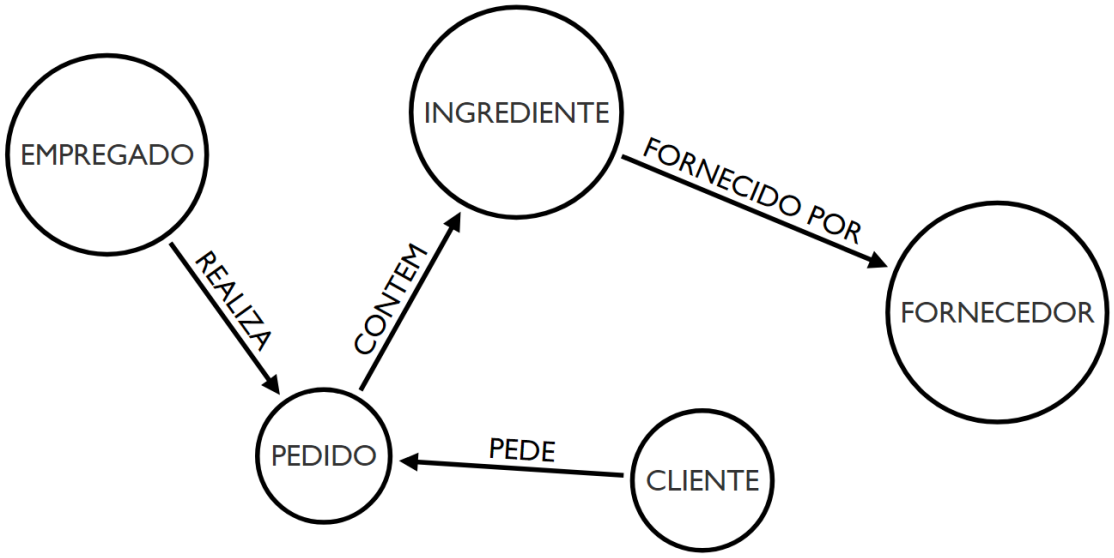


Fig 35: Estrutura base para o sistema NoSQL.

Foram criados os nodos:

- ★ “Empregado” - cada nodo contém informação como “Id”, “Nome” e “Telefone”
- ★ “Pedido” - tem como atributos o “Id_cliente”, “Id_empregado”, ”Preco” e “Data” em que “Data” está dividida num array em que índice 0 corresponde ao ano, índice 1 corresponde ao mês e índice 2 corresponde ao dia;
- ★ “Cliente” - tem como atributos o “Nr_contribuinte” e ”Nome”,
- ★ “Ingrediente” - contém “Id”, “nome”, “Stock”, “Id_fornecedor”, “PrecoCompra” e “PrecoVenda”;
- ★ “Fornecedor” - contém “Nome” e “Id”.

As relações identificadas entre os nodos são: entre “Empregado” e “Pedido” existe uma relação “Realiza”, em que empregado realiza um ou mais pedidos. Existe outra relação entre “Cliente” e “Pedido” designada por “Pede”, no qual um cliente pede um pedido. Por fim os nodos “Pedido” e “Ingrediente” têm a relação “Contem”, em que um pedido contém um ou mais ingredientes. Por último “Ingrediente” e “Fornecedor” têm uma relação designada por “Fornecido por”, no qual um ingrediente é fornecido por um determinado fornecedor.

6.5 Identificação dos objetos de dados SQL que serão utilizados no novo sistema

Os objetos de dados SQL que serão utilizados neste novo sistema são apenas os necessários para responder aos requisitos e interrogações propostas, tais como as tabelas

“Cliente”, “Empregado”, “Ingrediente”, “Pedido” e “Fornecedor” que passam a nodos e a tabela “PedidoIngrediente” que passa a ser uma relação, chamada “Contem”, entre “Pedido” e “Ingrediente” no novo sistema.

6.6 Processo de migração de dados

O processo de migração de dados de um sistema SQL para um sistema NoSQL está dividido em várias fases mostradas a seguir.

6.6.1 Mapeamento e Processo de conversão

Para o processo de conversão do sistema SQL para o sistema NoSQL foi feito um mapeamento “direto”. Cada linha das tabelas “Empregado”, “Cliente”, “Pedido”, “Ingrediente” e “Fornecedor” do sistema SQL deu origem a um nodo no sistema NoSQL e a tabela “PedidoIngrediente” do sistema SQL deu origem à relação entre os nodos do tipo Pedido com os nodos do tipo Ingrediente.

6.6.2 Extração, Transformação e Carregamento

Para a extração foi feito o export de todas as tabelas necessárias para ficheiros CSV com o respetivo nome da tabela. O export foi realizado através do MySQL workbench.

A transformação foi realizada com a criação de um ficheiro CYPHER que lê ficheiros CSV e com os dados lidos consegue criar nodos e relações entre nodos.

Para carregar os dados foi utilizado um ficheiro CYPHER que acede aos ficheiros CSV para fazer o load de todos dados guardados nos respetivos ficheiros e respetivas relações entre os vários nodos criados.

6.6.3 Apresentação e Descrição da implementação

Extração

- ★ SELECT * FROM Cliente;
- ★ SELECT * FROM Pedido;
- ★ SELECT * FROM Ingrediente;
- ★ SELECT * FROM Fornecedor;
- ★ SELECT * FROM Empregado;
- ★ SELECT * FROM PedidoIngrediente.

Transformação e Carregamento

Nodos

★ Cliente

```
LOAD CSV WITH HEADERS FROM 'file:///Cliente.csv' AS row
CREATE (c:Cliente {Nr_contribuinte:toInteger(row.Nr_contribuinte), Nome:toString(row.Nome)})
RETURN c;
```

★ Pedido

```
LOAD CSV WITH HEADERS FROM 'file:///Pedido.csv' AS row
CREATE (p: Pedido {Id: toInteger(row.Id)})
SET     p.Id_cliente = toInteger(row.Id_cliente),
        p.Id_empregado = toInteger(row.Id_empregado),
        p.Preco = toFloat(row.Preco),
        p.Data = split(row.Data, '-')
RETURN p;
```

★ Ingrediente

```
LOAD CSV WITH HEADERS FROM 'file:///Ingrediente.csv' AS row
CREATE (i: Ingrediente {nome: toString(row.Nome)})
SET     i.Id = toInteger(row.Id),
        i.Stock = toInteger(row.Stock),
        i.Id_fornecedor = toInteger(row.Id_fornecedor),
        i.PrecoCompra = toFloat(row.PrecoCompra),
        i.PrecoVenda = toFloat(row.PrecoVenda)
RETURN i;
```

★ Fornecedor

```
LOAD CSV WITH HEADERS FROM 'file:///Fornecedor.csv' AS row
CREATE (f: Fornecedor {Id: toInteger(row.Id), Nome: toString(row.Nome)})
RETURN f;
```

★ Empregado

```
LOAD CSV WITH HEADERS FROM 'file:///Empregado.csv' AS row
CREATE (e: Empregado {Id: toInteger(row.Id)})
SET e.Nome = toString(row.Nome),
        e.Telefone = toString(row.Telefone)
RETURN e;
```

Relações

★ Cliente - Pedido

```
MATCH (c: Cliente),(p: Pedido)
WHERE c.Nr_contribuinte = p.Id_cliente
CREATE (c)-[:PEDE]->(p);
```

★ Empregado - Pedido

```
MATCH (e: Empregado),(p: Pedido)
WHERE e.Id = p.Id_empregado
CREATE (e)-[:REALIZA]->(p);
```

★ Pedido - Ingrediente

```
LOAD CSV WITH HEADERS FROM 'file:///PedidoIngrediente.csv' AS row
MATCH (p: Pedido),(i: Ingrediente)
WHERE p.Id = toInteger(row.Id_pedido) AND toInteger(row.Id_ingrediente) = i.Id
CREATE (p)-[r:CONTEM {Preco: toFloat(row.Preco), Quantidade: toInt(row.Quantidade)}]->(i);
```

★ Ingrediente - Fornecedor

```
MATCH (i: Ingrediente),(f: Fornecedor)
WHERE i.Id_fornecedor = f.Id
CREATE (i)-[:FORNECIDO_POR]->(f);
```

6.7 Linguagem de interrogação do sistema NoSQL para as interrogações realizadas

★ **Top 5 ingredientes mais pedidos, com respectivo fornecedor**

```
MATCH (p:Pedido)-[r:CONTEM]->(i: Ingrediente)-[:FORNECIDO_POR]->(f:Fornecedor)
RETURN i.nome AS Designacao, f.Nome,COUNT(i.Id)
ORDER BY COUNT(i.Id) DESC
LIMIT 5;
```

★ **Qual o cliente que mais gastou**

```
MATCH (c: Cliente)-[:PEDE]->(p: Pedido)
RETURN c.Nome, SUM(p.Preco)
ORDER BY SUM(p.Preco) DESC
LIMIT 1;
```

★ **Qual o cliente com mais pedidos**

```
MATCH (c: Cliente)-[:PEDE]->(p: Pedido)
RETURN c.Nome, COUNT(p.Id_cliente)
ORDER BY COUNT(p.Id_cliente) DESC
LIMIT 1;
```

★ **Qual o empregado que efetuou mais pedidos**

```
MATCH (e: Empregado)-[:REALIZA]->(p: Pedido)
RETURN e.Nome, COUNT(p.Id_empregado)
ORDER BY COUNT(p.Id_empregado) DESC
LIMIT 1;
```

★ **Quantas vezes um cliente pediu um dado ingrediente distribuído por um determinado fornecedor**

```
MATCH (c:Cliente)-[:PEDE]->(p:Pedido)-[:CONTEM]->(i:Ingrediente)-[:FORNECIDO_POR] ->(f:Fornecedor)
WHERE c.Nr_contribuinte = 54321 AND i.nome = "Vinho" AND f.Nome = "Refrigerantes do Norte"
RETURN i.nome AS Designacao,COUNT(i.Id) AS Vezes_que_pediu, f.Nome AS Fornecedor;
```

★ **Qual o lucro da pizzaria num dado intervalo de tempo**

```
MATCH (p: Pedido)-[r:CONTEM]->(i: Ingrediente)
WHERE p.Data[0] = "2018" AND p.Data[1] = "11"
RETURN SUM((i.PrecoVenda - i.PrecoCompra) * r.Quantidade) AS LUCRO;
```

Conclusão Etapa 1

No final deste trabalho, o grupo considera que conseguiu aplicar de forma correta o que foi lecionado ao longo do semestre, relativamente ao desenvolvimento de uma base de dados relacional. Foi-nos transmitido o conceito de base de dados, qual a sua funcionalidade, quais as vantagens da sua utilização, formas de gerir a informação nela contida (codificando triggers, transactions, views, functions, procedures, etc.). A partir daqui o grupo refletiu e chegou a um consenso em relação ao tema a escolher, base de dados de uma pizzaria. Este é um tema que, considera o grupo, é suficiente para serem aplicados os conceitos que nos foram sendo transmitidos.

Com o desenvolvimento do MC e do ML, reparou-se na importância que é preciso dar às funcionalidades, neste caso, da pizzaria, visto que é necessário desenvolver e especificar de forma muito concreta e concisa os requisitos da base de dados desenvolvida. O grupo teve o cuidado de, desde o início, construir ambos os modelos de forma a cumprirem tanto os requisitos como as formas normais, respetivamente.

Em relação ao modelo físico, o professor que leciona as aulas teóricas desta unidade curricular ensinou-nos da melhor maneira os principais comandos para trabalharmos com as tabelas. A partir disto foi mais acessível gerarmos o modelo físico, algo conseguido também com a utilização da funcionalidade *forward engineering*, disponibilizada pelo *MySQL*.

Todos os elementos do grupo enriqueceram o seu conhecimento e cresceram, tanto pessoalmente como academicamente, pois fomentam o espírito de crítica, em relação ao seu trabalho e ao dos colegas, como também abordaram certos conceitos que até então eram desconhecidos.

O trabalho apresentado foi construído com o intuito de ser o mais legível possível, com identificação correta das entidades e respetivos atributos e com *queries* codificadas de modo a poder ser feita uma boa exploração da base de dados.

Conclusão Etapa 2

No final desta etapa 2, todos os nossos objetivos foram cumpridos.

Conseguimos fazer a migração com sucesso de uma base de dados relacional para uma base de dados não relacional, tal como pretendido e entender certas vantagens que estas bases de dados NoSQL podem vir a trazer, tais como a fácil manutenção destas e o desempenho das queries.

Enriquecemos o nosso conhecimento no que toca a bases de dados orientadas a grafos, como o Neo4j, e que por vezes estas podem ser fiáveis, mas tudo o que traz vantagens traz também desvantagens. No nosso trabalho estas desvantagens acabam por não ser tão notáveis pois trata-se de uma base de dados pequena e de fácil de manutenção, mas sabemos que elas existem. Também achamos interessante que, ao ser possível visualizar a base de dados como um grafo, a sua “travessia” para extração de informação se torna muito mais intuitiva.

Como consequência do estudo realizado, concluímos que quando se pretende construir uma base de dados é necessária atenção especial para aspectos importantes como, por exemplo: o seu crescimento, ou seja, se esta vai crescer exponencialmente ou não; se a estrutura não é algo importante; se o foco está no tempo de resposta de interrogações. Estes aspectos são fundamentais ter em conta na hora de decisão da implementação de uma base de dados, pois acabam por decidir se será mais vantajoso a utilização de um sistema SQL ou NoSQL.

Análise Crítica Etapa 1

Ao fazer uma análise ao trabalho realizado pelos elementos do grupo, estes acham que o que foi desenvolvido corresponde aos requisitos propostos para esta primeira parte do trabalho de Bases de Dados (desenvolver uma base de dados relacional). Foram aplicados corretamente os conceitos de entidades, atributos, chaves, relacionamentos, por forma a representar os dados da pizzaria da maneira mais correta possível.

Contudo, existem certos pormenores que podiam ser melhorados, como por exemplo, a gestão do stock dos ingredientes da pizzaria. Estes estão a ser colocados nos pedidos de uma forma não especificada, mas poderiam ser escolhidos de acordo com a data de validade, bastando para isso adicionar um atributo, com o tipo *DATE*, à tabela 'Ingrediente'. Fazendo com que a tabela 'Ingrediente' fique com uma chave primária composta.

Análise Crítica Etapa 2

Na realização da segunda etapa do trabalho prático, o grupo implementou uma base de dados NoSQL, utilizando o Neo4j, um sistema de base de dados orientado a grafos. O grupo definiu os nodos de forma a aproveitar da melhor maneira os benefícios de uma estrutura definida por nodos. Alguns atributos não foram migrados para a nova base de dados, atributos especificados na implementação da etapa anterior, pois o grupo entendeu que para as interrogações a serem desenvolvidas, os mesmos não seriam necessários.

O grupo poderia ter definido interrogações mais complexas, de modo a alargar as funcionalidades da base de dados NoSQL, mas no âmbito do tema estudado para o projeto, tal não foi necessário.

Bibliografia

[1] T. Carolyn, E.Begg, Database Systems, 4th Edition ed.

[2] “MySQL 8.0 Reference Manual :: 11.8 Data Type Storage Requirements,” [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html?fbclid=IwAR0cTA-WNuuRE4iX1pwOVU4j5YsBoSCJRf0HIg1hFM7st3oPvWWufD4XNt8.

[3] Robinson, Ian, Jim Webber, and Emil Eifrem. *Graph databases*. " O'Reilly Media, Inc.", 2015.

Lista de Siglas e Acrónimos

BD Base de Dados

SBD Sistema de Bases de Dados

PZdC Pizzaria Zé do Cartaxo

ER Entidade-Relacionamento

ML Modelo Lógico

MC Modelo Conceptual

Anexos

I- Anexo 1 - Email enviado pelo Sr. Zé do Cartaxo

Bom dia Bruno Veloso, Jaime Leite, João Marques e Nuno Rei.

Venho por este meio comunicar toda a minha satisfação no trabalho realizado pelo grupo, as vendas aumentaram, o tempo de serviço reduziu bastante, deixou de haver erros nas informações sobre os clientes e na criação dos mesmos como duplicação de informação. Com este sistema consigo controlar melhor o stock dos meus ingredientes fazendo uma gestão mais inteligente dos mesmos.

Para eventos futuros que possam ocorrer serão a minha primeira escolha para a implementação de um novo sistema para um novo negócio que possa vir a criar com o crescimento e reconhecimento que tenho tido.

Muito obrigado a todos os membros do grupo, os meus cumprimentos e que o grupo de trabalho continue a fazer um excelente trabalho!

II- Anexo 2 – Script de criação da BD

```
CREATE TABLE IF NOT EXISTS `Pizzaria`.`Cliente` (
  `Nr_contribuinte` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `DataNascimento` DATE NOT NULL,
  `Localidade` VARCHAR(20) NOT NULL,
  `Email` VARCHAR(45) NULL,
  PRIMARY KEY (`Nr_contribuinte`))
ENGINE = InnoDB;
```

Fig 36: Criação da tabela Cliente.

```
CREATE TABLE IF NOT EXISTS `Pizzaria`.`ClienteContacto` (
  `Id_cliente` INT NOT NULL,
  `Telefone` VARCHAR(9) NOT NULL,
  PRIMARY KEY (`Id_cliente`, `Telefone`),
  CONSTRAINT `fk_CLienteContacto_Cliente1`
    FOREIGN KEY (`Id_cliente`)
    REFERENCES `Pizzaria`.`Cliente` (`Nr_contribuinte`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Fig 37: Criação da tabela ClienteContacto.

```
CREATE TABLE IF NOT EXISTS `Pizzaria`.`Empregado` (
  `Id` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Email` VARCHAR(45) NULL,
  `Telefone` VARCHAR(9) NOT NULL,
  `NIF` INT NOT NULL,
  `Localidade` VARCHAR(20) NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;
```

Fig 38: Criação da tabela Empregado.

```
CREATE TABLE IF NOT EXISTS `Pizzaria`.`Fornecedor` (
  `Id` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;
```

Fig 39: Criação da tabela Fornecedor.

```
CREATE TABLE IF NOT EXISTS `Pizzaria`.`Ingrediente` (
  `Id` INT NOT NULL,
  `Nome` VARCHAR(45) NOT NULL,
  `Stock` INT NOT NULL,
  `PrecoCompra` DECIMAL(8,2) NOT NULL,
  `PrecoVenda` DECIMAL(8,2) NOT NULL,
  `Id_fornecedor` INT NOT NULL,
  PRIMARY KEY (`Id`),
  INDEX `fk_Ingrediente_Fornecedor1_idx` (`Id_fornecedor` ASC),
  CONSTRAINT `fk_Ingrediente_Fornecedor1`
    FOREIGN KEY (`Id_fornecedor`)
    REFERENCES `Pizzaria`.`Fornecedor` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Fig 40: Criação da tabela Ingrediente.

```
CREATE TABLE IF NOT EXISTS `Pizzaria`.`Pedido` (
  `Id` INT NOT NULL,
  `Id_cliente` INT NOT NULL,
  `Id_empregado` INT NOT NULL,
  `Preco` DECIMAL(8,2) NOT NULL,
  `Data` DATE NOT NULL,
  PRIMARY KEY (`Id`),
  INDEX `fk_Pedido_Cliente1_idx` (`Id_cliente` ASC),
  INDEX `fk_Pedido_Empregado1_idx` (`Id_empregado` ASC),
  CONSTRAINT `fk_Pedido_Cliente1`
    FOREIGN KEY (`Id_cliente`)
    REFERENCES `Pizzaria`.`Cliente` (`Nr_contribuinte`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Pedido_Empregado1`
    FOREIGN KEY (`Id_empregado`)
    REFERENCES `Pizzaria`.`Empregado` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Fig 41: Criação da tabela Pedido.

```
CREATE TABLE IF NOT EXISTS `Pizzaria`.`PedidoIngrediente` (
  `Id_pedido` INT NOT NULL,
  `Id_ingrediente` INT NOT NULL,
  `Preco` DECIMAL(8,2) NOT NULL,
  `Quantidade` INT NOT NULL,
  PRIMARY KEY (`Id_pedido`, `Id_ingrediente`),
  INDEX `fk_PedidoIngrediente_Ingrediente1_idx` (`Id_ingrediente` ASC),
  CONSTRAINT `fk_PedidoIngrediente_Ingrediente1`
    FOREIGN KEY (`Id_ingrediente`)
    REFERENCES `Pizzaria`.`Ingrediente` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_PedidoIngrediente_Pedido1`
    FOREIGN KEY (`Id_pedido`)
    REFERENCES `Pizzaria`.`Pedido` (`Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Fig 42: Criação da tabela PedidoIngrediente.

III- Anexo 3 – Script do povoamento da BD

```
INSERT INTO Empregado
    (Id, Nome, Email, Telefone, NIF, Localidade)
    VALUES
        (1, 'Antunes' , null , '919191919' , 13579 , 'Porto'),
        (2, 'Zé Cartaxo' , 'PZdC@gmail.com' , '969696969' , 24680 , 'Braga');
```

Fig 43: Povoamento da tabela Empregado.

```
INSERT INTO Fornecedor
    (Id, Nome)
    VALUES
        (1, 'Legumes Frescos'),
        (2, 'Conservas PT'),
        (3, 'Farinhas Portugal'),
        (4, 'Refrigerantes do Norte'),
        (5, 'Águas Fritalis');
```

Fig 44: Povoamento da tabela Fornecedor.

```
INSERT INTO Ingrediente
    (Id, Nome, Stock, PrecoCompra, PrecoVenda, Id_fornecedor)
    VALUES
        (1, 'Farinha Trigo' , 20 , 0.30 , 0.60 , 3 ),
        (2, 'Farinha Mista' , 20 , 0.33 , 0.66 , 3 ),
        (3, 'Farinha Integral' , 20 , 0.35 , 0.70 , 3 ),
        (4, 'Azeitonas' , 13 , 0.99 , 1.98 , 2 ),
        (5, 'Fiambre' , 40 , 0.45 , 0.90 , 2),
        (6, 'Queijo' , 50 , 0.39 , 0.78 , 2),
        (7, 'Ananás' , 3 , 0.95 , 1.90 , 1 ),
        (8, 'Tomate' , 10 , 0.40 , 0.80 , 1 ),
        (9, 'Pimento' , 5 , 1.20 , 2.40 , 1 ),
        (10, 'Cebola' , 6 , 0.74 , 1.48 ,1 ),
        (11, 'Cogumelos' , 9 , 0.43 , 0.86 , 2 ),
        (12, 'Coca Cola' , 15 , 1.00 , 2.00 , 4 ),
        (13, 'Água' , 15 , 0.70 , 1.40 , 5 ),
        (14, 'Água Pedras' , 10 , 0.90 , 1.80 , 5 ),
        (15, 'Cerveja' , 8 , 1.20 , 2.40 , 4 ),
        (16, 'Ice Tea' , 12 , 1.00 , 2.00 , 4 ),
        (17 , 'Vinho' , 1  , 5.00 , 10.00 , 4);
```

Fig 45: Povoamento da tabela Ingrediente.

```sql
INSERT INTO Cliente
    (Nr_contribuinte, Nome, DataNascimento, Localidade, Email)
    VALUES
        (81826 , 'Marques' , '1985-12-25' , 'Braga' , 'a81826@alunos.uminho.pt'),
        (80757 , 'Jaime' , '1956-11-07' , 'Braga' , 'a80757@alunos.uminho.pt'),
        (81918 , 'Rei' , '1989-05-27' , 'Braga' , NULL),
        (78352 , 'Bruno' , '1980-04-20' , 'Braga' , 'a78352@alunos.uminho.pt'),
        (12345 , 'António' , '2000-01-11' , 'Porto' , 'antoninho@gmail.com'),
        (54321 , 'Joana' , '1960-05-12' , 'Guimarães' , 'joaninha@hotmail.com'),
        (98765 , 'Rafael' , '1995-10-06' , 'Porto' , 'rafa@portugalmail.pt');
```

Fig 46: Povoamento da tabela Cliente.

```sql
INSERT INTO ClienteContacto
    (Id_cliente, Telefone)
    VALUES
        (81826,'912345678'),
        (80757,'987654321'),
        (81826,'911111111'),
        (81918,'123456789');
```

Fig 47: Povoamento da tabela ClienteContacto.

```sql
INSERT INTO Pedido
    (Id, Id_cliente, Id_empregado, Preco, Data)
    VALUES
        (1 , 81826 , 1 , 9.86 , '2018-11-11'),
        (2 , 81918 , 2 , 12.90 ,'2018-11-11'),
        (3 , 80757 , 1 , 7.20 , '2018-11-13'),
        (4 , 81826 , 2 , 6.36 , '2018-11-14'),
        (5 , 78352 , 2 , 7.54 , '2018-11-15'),
        (6 , 81918 , 2 , 9.38 , '2018-11-15'),
        (7 , 81826 , 1 , 14.82 , '2018-11-15'),
        (8 , 54321 , 1 , 15.5 , '2018-11-16'),
        (9 , 98765 , 2 , 9.06 , '2018-11-18'),
        (10 , 12345 , 1 , 9.44 , '2018-12-20'),
        (11 , 98765 , 2 , 8.94 , '2019-01-9');
```

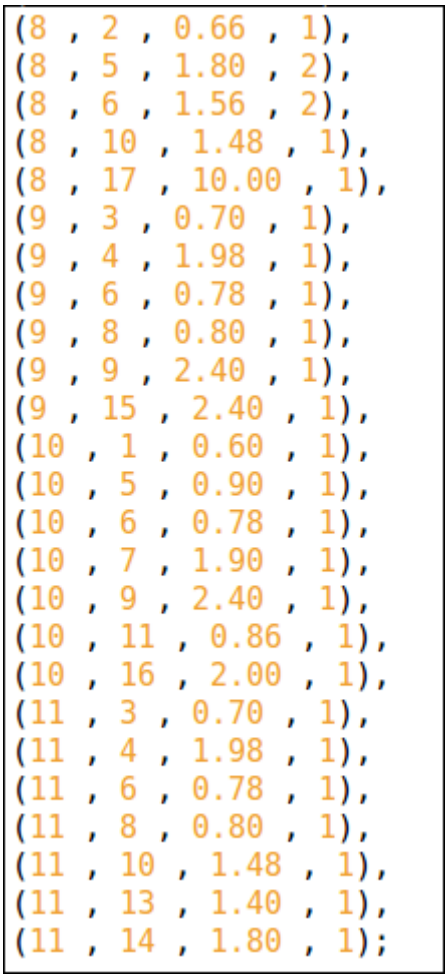
Fig 48: Povoamento da tabela Pedido.

```
INSERT INTO PedidoIngrediente
    (Id_pedido, Id_ingrediente, Preco, Quantidade)
    VALUES
        (1 , 8 , 0.80 , 1),
        (1 , 12 , 4.00 , 2),
        (1 , 11 , 0.86 , 1),
        (1 , 2 , 0.66 , 1),
        (1 , 4 , 1.98 , 1),
        (1 , 6 , 1.56 , 2),
        (2 , 7 , 1.90 , 1),
        (2 , 8 , 0.80 , 1),
        (2 , 9 , 2.40 , 1),
        (2 , 10 , 1.48 , 1),
        (2 , 11 , 0.86 , 1),
        (2 , 16 , 2.00 , 1),
        (2 , 3 , 0.70 , 1),
        (2 , 4 , 1.98 , 1),
        (2 , 6 , 0.78 , 1),
        (3 , 12 , 4.00 , 2),
        (3 , 13 , 1.40 , 1),
        (3 , 14 , 1.80 , 1),
        (4 , 16 , 2.00 , 1),
        (4 , 3 , 0.70 , 1),
        (4 , 4 , 1.98 , 1),
        (4 , 5 , 0.90 , 1),
        (4 , 6 , 0.78 , 1),
```

Fig 49: Povoamento da tabela PedidoIngrediente.

```
(5 , 7 , 1.90 , 1),
(5 , 11 , 0.86 , 1),
(5 , 15 , 2.40 , 1),
(5 , 3 , 0.70 , 1),
(5 , 5 , 0.90 , 1),
(5 , 6 , 0.78 , 1),
(6 , 7 , 1.90 , 1),
(6 , 8 , 0.80 , 1),
(6 , 11 , 0.86 , 1),
(6 , 1 , 0.60 , 1),
(6 , 2 , 0.66 , 1),
(6 , 4 , 1.98 , 1),
(6 , 5 , 1.80 , 2),
(6 , 6 , 0.78 , 1),
(7 , 9 , 2.40 , 1),
(7 , 10 , 1.48 , 1),
(7 , 13 , 1.40 , 1),
(7 , 14 , 1.80 , 1),
(7 , 15 , 2.40 , 1),
(7 , 1 , 0.60 , 1),
(7 , 4 , 3.96 , 2),
(7 , 6 , 0.78 , 1),
```

Fig 50: Povoamento da tabela PedidoIngrediente (Continuação).



```
(8 , 2 , 0.66 , 1),
(8 , 5 , 1.80 , 2),
(8 , 6 , 1.56 , 2),
(8 , 10 , 1.48 , 1),
(8 , 17 , 10.00 , 1),
(9 , 3 , 0.70 , 1),
(9 , 4 , 1.98 , 1),
(9 , 6 , 0.78 , 1),
(9 , 8 , 0.80 , 1),
(9 , 9 , 2.40 , 1),
(9 , 15 , 2.40 , 1),
(10 , 1 , 0.60 , 1),
(10 , 5 , 0.90 , 1),
(10 , 6 , 0.78 , 1),
(10 , 7 , 1.90 , 1),
(10 , 9 , 2.40 , 1),
(10 , 11 , 0.86 , 1),
(10 , 16 , 2.00 , 1),
(11 , 3 , 0.70 , 1),
(11 , 4 , 1.98 , 1),
(11 , 6 , 0.78 , 1),
(11 , 8 , 0.80 , 1),
(11 , 10 , 1.48 , 1),
(11 , 13 , 1.40 , 1),
(11 , 14 , 1.80 , 1);
```

Fig 51: Povoamento da tabela PedidoIngrediente (Continuação).

IV- Anexo 4 – Function temStock

```
DELIMITER $$
CREATE FUNCTION temStock (ing INT)
    RETURNS CHAR(3)
    READS SQL DATA
    DETERMINISTIC
BEGIN
    DECLARE res CHAR(3);
    SET res = (SELECT IF (I.Stock != 0, I.Stock, 0)
                    FROM Ingrediente AS I
                    WHERE I.Id = ing);
    RETURN res;
END; $$
DELIMITER ;
```

Fig 52: Function que retorna a quantidade de stock de um ingrediente.